4.12-04 - CQ - Entrega Desafio 01

Victor Puntel Rui
Gustavo Alves Beneti

Introdução

Este trabalho foi realizado na disciplina de cinética química com o intuito de simular um sistema ideal que obedeça as leis físicas.
Para que a simulação seja realizada de forma ideal, utilizamos conceitos da física como; distribuição de Maxwell-Boltzman; conservação de energia; colisões e dentre outras

Metodologia

Como citado anteriormente, para realizar a atividade, combinamos a teoria física de modelagem de sistemas, com a implementação computacional.
Este tópico de Metodologia será dividido em duas partes. Uma que descreve o comportamento físico e as leis que regem o sistema. A outra, descreverá a implementação no código em si.

Física do sistema

Conservação de Energia

"*Na natureza, nada se cria, nada se perde, tudo se transforma*"
-Antoine-Laurent de Lavoisier

Este, talvez, seja o conceito mais fundamental e importante.
Quando falamos de um sistema fechado, a primeira coisa que devemos pensar é a conservação de energia do sistema. Não podemos criar energia do nada e nem retira-la de forma alguma, ela apenas se transforma, transitando entre suas diversas formas; energia cinética, mecânica, térmica, química, nuclear e dentre outras. Entretanto, como estamos falando de um sistema a nível atômico simples, vamos trabalhar somente com a conservação da energia cinética. Futuramente nos próximos desafios podemos implementar novas formas de energia.

Distribuição de Maxwell - Boltzmann

Para a simulação ficar mais realista o possível, temos que aplicar leis físicas que ja forma observadas em um sistema real. A distribuição de Maxwell-Boltzmann é uma distribuição de probabilidade que têm aplicações tanto na física quanto na química.

As velocidades das partículas presentes em um gás ideal obedecem uma distribuição bem definida. Se nossa simulação obedecer esta distribuição, então teremos certeza que ela estará a um passo a mais de um sistema ideal.

Colisões

Como estamos trabalhando com um conjunto de partículas se movendo em um espaço delimitado, o fator de colisão irá ocorrer constantemente e é crucial que seja calculado corretamente.
Uma colisão envolve fatores como; velocidade da partícula; ângulo de contato; massa dos corpos; vetor de deslocamento.
Para que as colisões sejam realizadas da forma correta, também temos que pensar na conservação da energia cinética do sistema como um todo, de forma que cada colisão conserve sua energia e momento.

Conservação de momento

Dito isso, temos também a lei de conservação de momento após as colisões. Portanto, a totalidade de momento (massa e velocidade) do sistema como um todo, não muda. Como a massa de cada partícula não muda após a colisão, somente nos resta a possibilidade de a velocidade ser alterada, sendo assim, a soma das velocidades de todas as partículas, tem que ser igual em qualquer tempo T da simulação.
É ai que entra a distribuição de Maxwell-Boltzmann, as partículas vão colidindo de tal forma que, a distribuição de velocidades destas, se comporta da forma que a distribuição descreve.

Implementação

Classes

Com o fim de deixar o código mais otimizado e organizado, optamos por utilizar as classes.
As classes do python, têm uma organização própria e é a forma mais indicada de dar características a objetos, sem criar listas e matrizes recheadas de valores, quando tratamos de objetos com características, propriedades e comportamentos semelhantes.

Colisão em código e vetorização

A implementação da colisão no código foi algo complexo e que gastamos a maior parte do tempo do projeto. Quando tentávamos realizar a simulação, as partículas muitas vezes não colidiam, ou se grudavam e até mesmo aceleravam bruscamente após o contato entre os dois corpos.

O modelo que atendeu as demandas teve como base principalmente:

- A utilização da biblioteca matplotlib para a conversão das velocidades para vetores
- A criação de uma função que checa se ocorreu colisão ou não, através da distância entre um raio e outro de uma partícula.

```
def check_collision(p1, p2):
    #checa a colisao por distancia euclidiana e soma dos raios
    distance = math.sqrt((p2.x - p1.x)**2 + (p2.y - p1.y)**2)
    if distance <= p1.raio + p2.raio: #checa
        return True #colide
    else:
        return False #nao colide
```

Como podemos ver aqui em um trecho do código, apenas com fim de exemplificar o dito acima, caso a distância entre os raios de duas partículas seja menor ou igual que a soma de ambos, a colisão é detectada.

Usamos a equação (eq 1) como modelo para colisão. O motivo foi principalmente por conta de que ela já está escrita em vetor.

$$v_1' = v_1 - \frac{2m_2}{m_1 + m_2} \frac{\langle v_1 - v_2, x_1 - x_2 \rangle}{||x_1 - x_2||^2}(x_1 - x_2)$$

$$v_2' = v_2 - \frac{2m_1}{m_1 + m_2} \frac{\langle v_2 - v_1, x_2 - x_1 \rangle}{||x_2 - x_1||^2}(x_2 - x_1)$$

Equação retirada da referência (1)

Resultados

Histograma da distribuição de velocidades

Conforme o esperado, foi obtida uma simulação que obedece a distribuição de velocidade de Maxwell-Boltzmann, como podemos observar na figura 2, onde temos no eixo X a velocidade das partículas e, no Y, a frequência de partículas que aparecem com esta velocidade.

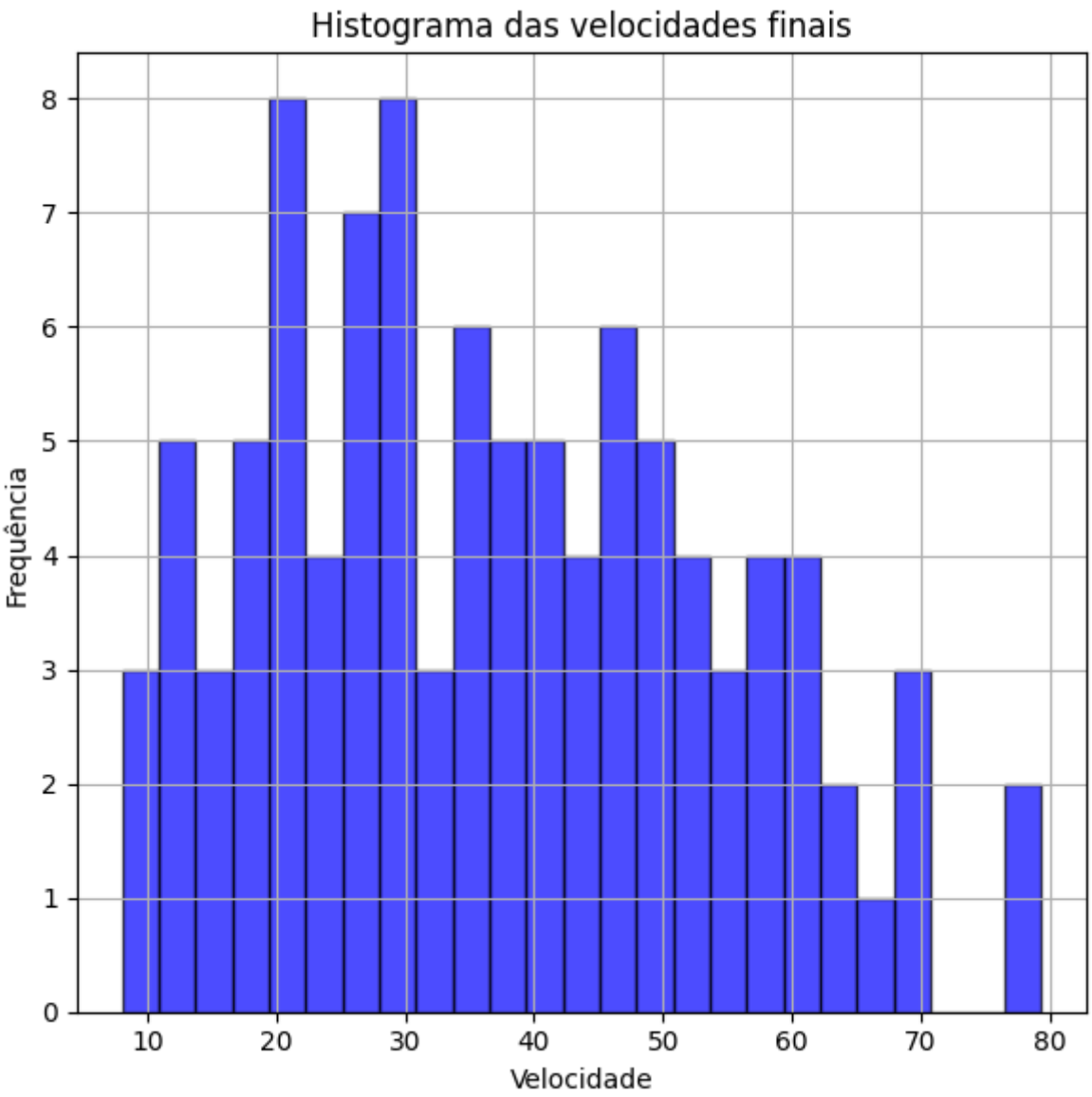

Figura 2

Quanto a nossa visualização, conseguimos criar o display que mostra como elas estão distribuídas no espaço. Para tal, usamos o matplotlib.

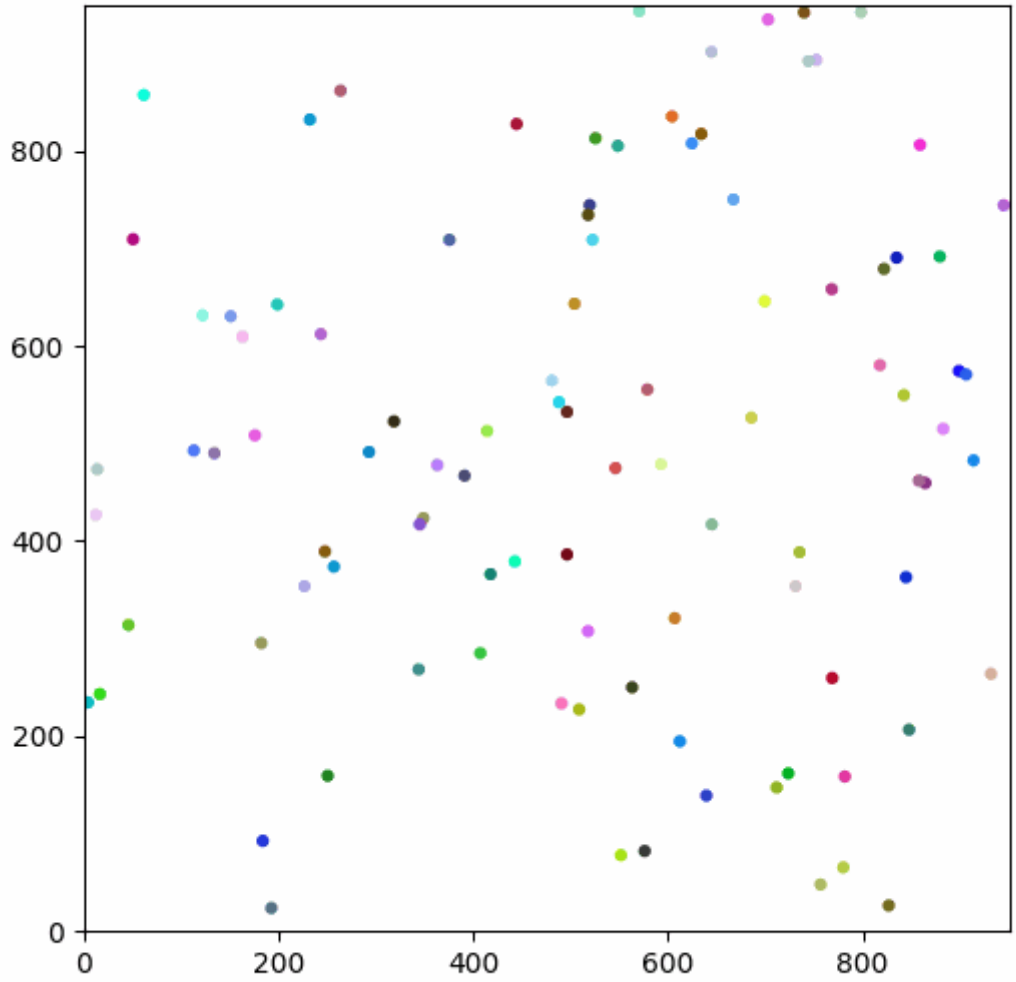

Figura 3

> **Info**
>
> Note que algumas partículas ainda estão grudadas. Futuramente tentaremos implementar uma condição na geração das partículas que as impeça de serem geradas no mesmo espaço que a outra.

Conclusão

Por fim, com a distribuição de Maxwell-Boltzmann funcionando, sabemos que a colisão e o sistema no geral está obedecendo as leis físicas de um sistema real, possibilitando assim o desenvolvimento futuro de simulações atomísticas mais complexas, visto que a base do sistema já está bem consolidada.

Referências

(1) Wikipédia - English - Elastic Colision: https://en.wikipedia.org/wiki/Elastic_collision#Two-dimensional_collision_with_two_moving_objects
(2) Donald A. McQuarrie; John D. Simon (1997). Physical Chemistry - *A molecular approach*

Disponibilidade do Código:

Nosso github para mais detalhes: https://github.com/VictorPuntelRui/cinetica_quim